Access Control Lists Design

Goal

Provide a simple version of the existing ECHO ACLs in the CMR with good performance, storage in Metadata DB, and support for editing from MMT.

Traceability

CMR-2218 - JIRA project doesn't exist or you don't have permission to view it.

ACLs Epic (Note, stories for the capabilities in this review have not yet been created.)

Background

Access Control Lists (ACLs) in ECHO went through several redesigns during the lifetime of ECHO. They were called Permissions originally. The current implementation (as of February 2016) was designed in October 2009 and subsequently implemented. It was a complete redesign and ground up rewrite that tried to address limitations and complexities of the older implementations.

ECHO_OpsCon_01...Roles).doc.pdf

The design was based on best practices from Role Based Security Best Practices (Ferraiolo, Kuhn, & Chandramouli, 2003). This was a good approach as it used well known best practices instead of reinventing the wheel. The ACLs in ECHO could be summarized as:

- Data and actions are restricted by default. You only gain access to data or invoke an endpoint through an ACL.
- ACLs are divided into two parts:
    - Access Control Entries (ACEs)
        - Determines *who* the ACL applies to and what *permission* (create, read, update, delete, order) they have.
        - These identify GUIDs of groups, all guest users, or all registered users.
    - Object Identity
        - Identifies *what* is being permitted, the nouns of the system. Providers, catalog items, groups, orders, etc.
        - Contains different levels of detail for identifying the object in ECHO depending on how much detail was needed. Catalog item permissions needed to be very fine grained to allow providers precise control for what items should be matched on a search response. Permissions to access orders were granted at a much broader level. Access to orders can be granted at an individual provider level or across the entire system. Groups were somewhere in the middle. They can be granted at a system level, provider level, or by identifying a specific group GUID.

Limitations with the Existing Design and Implementation

There are issues and limitations of the current implementation. Some of these we'd like to address as we move the implementation to the CMR. The existing SOAP API can be seen here: http://api.echo.nasa.gov/echo/ws/v10/AccessControlService.html

- Performance of fetching ACLs is slow. It currently takes about a minute to fetch ACLs from the ECHO REST API. The performance here was improved recently from 5 minutes but this is still too long.
- API
    - The API uses security based terms and acronyms (SID, access control entry) that aren't familiar to client developers not versed in role based security. There are equivalent more familiar terms for these concepts.
    - The ACL structure is overly nested. Clients creating an ACL or fetching data from ACL have to create many intermediate objects or navigate through this. The same information could be represented in a more flattened representation.

    - GUIDs are returned at every level of the ACL. Only the top level GUID is useful to clients. The internal GUIDs are used for for maintaining references across normalized table storage. This is an internal implementation detail that shouldn't be exposed on the API.
- Complex Storage
    - ACLs are stored normalized across 12 separate tables. Each table has an associated DAO implementation and interface. There's nothing specifically wrong with this approach but it's a lot of tables and code to maintain for storing a relatively small amount of data. Adding a new ACL type or field requires adding new columns or tables.

MMT Needs

MMT will be replacing PUMP as a client for administrators and providers to use for manipulating permissions and groups. MMT has mock ups for the permission pages at the following links. You must be logged in to view them. (If it redirects to the home page you need to logn

- https://mmt.sit.earthdata.nasa.gov/permissions
- https://mmt.sit.earthdata.nasa.gov/new-permissions
- https://mmt.sit.earthdata.nasa.gov/show-permissions

The mock ups are based on design made a year ago which in turn were based on the existing capabilities in PUMP. MMT's needs as a client of ACLs can be summarized by the following:

- Allow fast retrieval of ACL information over an easy to use API preferably with a JSON response format.

Design Goals

The goals of the ACL service in the CMR are the following:

- **Utilize the strong points of the current design.**
    - We will add a new concept-agnostic capability to Metadata DB to decouple Access Control concepts from Metadata DB's implementation. To do this we will store any incoming records with an unrecognized concept type (i.e. other than collections, granules, etc.) in a generic concept table.
- **Fast performance**. It should be possible to fetch all the ACLs in a matter of seconds and a single ACL in subsecond time. We do not need subsecond time for fetching all the ACLs as caching is sufficient for providing fast access.
- **Up to date caches**: If ACLs are persisted in Metadata DB we can listen for messages of when ACLs are updated and refresh the cache immediately. There won't be an hour delay from when an ACL is updated until when it take effect.
- **Backwards compatible.** The new ACL structure should be backwards compatible with the existing implementation. We will need to synchronize the data with ECHO for a period of time. We don't want to have to rewrite application of ACLs in the CMR.
- **Satisfy MMT's needs.**
- **Simple API** -
    - Use simple terms familiar to CMR providers and administrators.
    - Remove redundant data like GUIDs
    - Simple structure avoided deeply nested hierarchies.
    - Use REST and JSON consistent with the rest of the CMR.
- **Easy to implement and extend.**

Design

Simplified ACL Representation

This defines the structure of the ACL representation in the CMR. It is similar to the existing representation with some small changes for clarity and simplification. A JSON schema will be developed for this during implementation.

- **legacy_guid:** Used for synchronization with ECHO. We will eventually get rid of this field. ACLs will be identified in the CMR using a concept id.
- **group_permissions:** This is equivalent to the Access Control Entry (ACE) in ECHO.
    - **permissions:** A list containing any combination of these strings: create, read, update, delete, order
        - The "order" permission is a bit odd as you would not expect to see this in a traditional list of permissions. It's a carry over from ECHO for assigning order permissions to individual catalog items. It simplifies the mapping between representations to keep this the same.
    - **group_id or user_type:** A group permission will contain a group id (concept id of a group) or a user type which can be "registered" or "guest".
- ***Identities - one of the following will appear at the top level in the ACL to identify the permitted data.***
    - **system_identity:** Used for assigning permissions at the system level in the CMR like the ability to create or delete providers or ACLs themselves.
        - **target:** A string matching the existing strings used in ECHO - http://api.echo.nasa.gov/echo/ws/v10/SystemObject.html
    - **provider_identity:** Used for assigning permissions to things owned by a provider

        - **provider_id:** The short alpha numeric provider identifier (i.e. "LPDAAC_ECS")
        - **target:** A string matching the existing strings used in ECHO - http://api.echo.nasa.gov/echo/ws/v10/ProviderObject.html
    - **single_instance_identity:** Used for assigning permissions to a single instance of an object. ECHO only ever used this for groups. We will keep the same structure as it may be useful in the future for adding permissions to a single type of object.
        - **target_id:** The identifier of the item (group concept id)
        - **target:** A string matching the existing strings used in ECHO - http://api.echo.nasa.gov/echo/ws/v10/SingleInstanceObject.html (only groups currently)
    - **catalog_item_identity:** Used for assigning permissions to groups of collections and granules. The structure of this map follows the structure in the SOAP documentation. fields below are not described in detail as their description is the same as originally.
        - **name:** The name of the catalog item identity. Catalog item identities have specific uses. Naming them allows providers to quickly identify what they're used for.
        - **provider_id**
        - **collection_applicable**
        - **granule_applicable**
        - **collection_identifier:** Optionally **i**dentifies controlled collections. All fields are optional. Collection id patterns and rolling temporal are not supported. These weren't used by providers and not supported in the CMR.
            - **entry_titles**
            - **access_value:** At least one of these must be specified. include_undefined_value or range values can be used but not together. If include_undefined_value is false a range must be specified.
                - **min_value**
                - **max_value**
                - **include_undefined_value**
            - **temporal:** No field as allowed in ECHO10. It's assumed to refer to acquisition date.
                - **start_date**
                - **stop_date**
                - **mask:** intersect, contains, or disjoint.
        - **granule_identifier**: Optionally identifies granules. Granule ur patterns and rolling temporal are not supported.
            - **access_value:** same structure and semantics as collection identifier
            - **temporal:** same structure and semantics as collection identifier

Referential Integrity

ACLs can refer to other things in the system:

- Groups
- Providers
- Collections through entry titles

Per feedback in external design review we will update the list of entry titles in this list when a collection is deleted. We'll need a user story for causing a collection delete to cascade to updating relevant ACLs. Based on comment from Edward Seiler We will also need to do the same thing when a group is deleted or a provider is deleted.

API

ACLs will be added to the new access control service in the CMR which currently houses groups.

- /acls
    - POST - Creates an ACL. Accepts JSON representation of an ACL. Returns concept id and revision id of the ACL.
    - GET - Search for ACLs
    - /:concept_id
        - GET - Gets an ACL.
        - PUT - Updates an ACL. Returns concept id and revision id of the ACL.
        - DELETE - Deletes an ACL. Returns concept id and revision id of the ACL tombstone.
    - /has_permission
        - GET - Checks if a given user has permission to identified data. See description below

Has Permission

Checks if a given user has permission to identified data.

Parameters

- **token:** User token either in header or in parameter list.
- **permission**: create, read, update, delete, or order of permission to check
- One of:
    - **system_identity_target:** The target of a system identity to check for access
    - **provider_id** and **provider_identity_target:** The provider id and target of a provider identity to check for access
    - **single_instance_identity_target** and **single_instance_identity_target_id:** The target id and target of a single instance identity to check for access.

        - **concept_id:** A collection or granule concept id to check for access.

ACL Search

Parameters:

- **Standard params**: page_num, page_size, pretty
- **permitted_group**: Can be a group concept id or a user type "registered" or "guest". Multiple values supported.
- **permission:** One or multiple of these strings: create, read, update, delete, order. Matches ACLs that grant those permissions.
- **provider**: Matches acls with a provider identity for the given provider. Multiple values supported.
- **identity_type**: String of "provider", "system", "single_instance", or "catalog_item" to identify ACLs of a particular type. Multiple values supported.
- **permitted_concept_id:** Search with a collection or granule concept id and matches ACLs that permit that collection or granule.
- **user:** A URS username. Matches ACLs that grant a permission to one of the groups to which the user belongs. This is similar to searching with multiple permitted groups.

Response Formats

Two response formats in JSON will be supported. We will identify the appropriate mime type with a vendor suffix
(like application/vnd.nasa.cmr.acl.ref+json

Reference

References contain a small amount of data for fast response times. MMT needs the ability to get a list of ACLs to display.

- Media Type: application/vnd.nasa.cmr.acl.ref+json
- URL Extension: .ref_json
- Fields per ACL
    - name - This will be the catalog item identity name or a string containing "<identity type> - <target>". For example "System - PROVIDER"
    - concept_id
    - revision_id
    - identity_type: String of "provider", "system", "single_instance", or "catalog_item"
    - location: URL to the ACL

Full ACLs

Returns the full ACL data in JSON.

- Media Type: application/vnd.nasa.cmr.acl+json
- URL Extension: .json

Questions

- What parameters will the MMT need to search for ACLs on?
    - Kathleen Carr will provide a list of the MMT stories to get the list of parameters. - provided spreadsheet to Jason and Mark on 3/10
- Does the reference response satisfy the MMT needs for retrieving permissions?
    - A: It should and we can add things if needed.
- Does MMT need the permitted_concept_id functionality?
    - A: We will just create a story for this capability and prioritize as needed.

Error rendering macro 'pageapproval' : null